DiG: Differential Grounding for Enhancing Fine-Grained Perception in Multimodal Large Language Models

Supplementary Material

6. Implementation Details

Training Configuration All experiments are conducted on the proposed Differential Grounding (DiG) dataset, which is procedurally generated through the 3D rendering pipeline introduced in Section 3.2. The corpus contains approximately 4.8K image pairs, evenly distributed across three subsets: single-difference, double-difference, and mixed-difference scenes (around 1.6K per subset). Each mixed-difference scene includes up to four visual discrepancies, while every image comprises up to ten distinct objects, ensuring a diverse range of visual layouts and attribute variations.

We employ Qwen3-VL-8B-Thinking and Qwen3-VL-4B-Thinking [2] as multimodal backbones. Reinforcement post-training is carried out using the EasyR1 framework [35, 53], optimized with the GRPO objective and KL-regularized policy optimization. Training follows the curriculum strategy outlined in Section 3.4, progressively increasing task complexity from single- to mixed-difference instances.

All hyperparameters, training schedules, and optimization settings are listed in Table 5.

Component	Parameter	Value
Algorithm	KL coefficient	1.0×10^{-2}
	Filter range	[0.01, 0.99]
	Reward weight α	0.1
Optimization	Learning rate	1.0×10^{-6}
	Weight decay	1.0×10^{-2}
	Optimizer	AdamW
	Gradient clipping	1.0
	Warmup ratio	0.0
Model	Gradient checkpointing	True
	Vision tower	Trainable
	FSDP sharding	Full
Rollout	n	5
	Temperature	1.0
	Top-p	1.0
	Validation override	T=0.6, p=0.95, n=1
Training Schedule	Stage 1 steps	60
	Stage 2 steps	20
	Stage 3 steps	30

Table 5. **Reinforcement Post-Training Configuration.** Key hyperparameters and rollout settings across the three curriculum stages.

To enable consistent instruction-following behavior during multimodal reasoning and grounding, we adopt a structured prompt format for the Differential Grounding (DiG) task. The prompt template we use is shown below.

Prompt for DiG Task

<image><image>You will be given two separate images.
- The first image is the 'before' version.
- The second image is the 'after' version.
Your task is to compare them and identify all 'changed regions'. Provide your answer as a JSON list of bounding boxes. All bounding boxes you provide must be on the **'before' (first) image**.
Follow these rules: 1. **Item is Missing or Changed:** If an item from the 'before' image is **missing** or **looks different** (e.g., color, shape, state) in the 'after' image, box the **original item on the 'before' image**. 2. **Item is New:** If a **new item appears** in the 'after' image (in a previously empty spot), box the **corresponding empty area on the 'before' image**.
Important: All coordinates must be relative to the **first ('before') image**. Use the format [x_1, y_1, x_2, y_2] with absolute pixel coordinates.

For Stage 1 and Stage 2, the model is additionally informed of the number of differences to facilitate curriculum-based reasoning.

Evaluation Configuration. Evaluation is conducted across a comprehensive suite of multimodal benchmarks, covering perception, grounding, and reasoning capabilities. The visual perception evaluation includes HalBench [14], HRBench-8K [41], POPE [24], V* [45], VSR [26], CV-Bench [39], and MMVP [40]. Grounding performance is assessed on RefCOCO, RefCOCO+ [20, 49], and RefCOCOg [30]. For multimodal reasoning and comprehension, we evaluate on MMBench [28], MM-Vet [50], MMStar [9], ScienceQA [29], TextVQA [36], MME [13], and AI2D [21].

Inference is conducted using the vLLM framework for efficient batched decoding across all benchmarks. Each evaluation employs batched inference with dynamic padding and a maximum generation length of 4096 tokens. The sampling temperature is fixed at 0, ensuring deterministic outputs across runs. During evaluation, the model generates both intermediate reasoning traces and final answers. For perception and general QA benchmarks, the model generates intermediate reasoning traces enclosed by the thinking tag. Following the "Think–Then–Answer" convention,

only the text after `</think>` is extracted and used as the final answer. In contrast, for grounding tasks, predictions are taken directly from the model's generated bounding boxes without post-hoc extraction. This extraction rule is consistently applied across all benchmarks to ensure comparable evaluation. All results reported in the main paper are obtained under these unified inference settings.

7. DiG Data Construction

To construct the Differential Grounding (DiG) dataset, we employ a controllable 3D scene generation pipeline that procedurally renders paired images under well-defined attribute configurations. Each scene consists of multiple objects parameterized by a set of compositional attributes, allowing precise manipulation of object-level differences across image pairs.

Attribute Definition. Objects are instantiated from four primary attribute groups: **shape**, **color**, **size**, and **material**. The *shape* attribute includes three geometric primitives—`Cube`, `Sphere`, and `Cylinder`—which form the core set of object templates. The *color* palette covers eight distinct hues (`blue`, `brown`, `cyan`, `gray`, `green`, `purple`, `red`, `yellow`) with fixed RGB coefficients to ensure visual consistency. Each object further takes one of three *size* levels (`small`, `medium`, `large`) corresponding to normalized scale factors of 0.4, 0.6, and 0.8, respectively. Finally, the *material* attribute controls surface reflectance through physically-based rendering parameters, where `metal` corresponds to ($\textit{metallic} = 1.0$, $\textit{roughness} = 0.2$) and `matte` to ($\textit{metallic} = 0.0$, $\textit{roughness} = 0.8$).

Scene Composition. For each scene, we randomly sample a subset of objects from the attribute space to populate a 3D layout. Controlled perturbations are then applied to introduce visual differences between paired images, such as changes in object shape, color, size, material, or count. This design enables a rich combination space while maintaining interpretable and disentangled factors of variation. The entire generation process is implemented directly in Python using Blender's rendering API, allowing precise reproducibility and randomized yet structured diversity across instances. Representative examples of the synthesized image pairs are illustrated in Fig. 7.

8. Analysis of Training Dynamics

Figure 8, 9, 10 illustrates the training dynamics of Qwen3-VL-8B-Thinking across the three curriculum stages. Overall, the entropy loss decreases steadily during the early stages and stabilizes as policy convergence occurs, indicating reduced exploration and more consistent generation behavior. The gradient norm remains bounded throughout training, confirming stable optimization. Both accuracy and format rewards exhibit monotonic improvement, with the latter quickly saturating near unity, suggesting the model rapidly adapts to the output structure constraints. Meanwhile, the average response length decreases as the model learns to produce more concise outputs, and CPU memory usage grows gradually with increased rollout complexity. Together, these trends demonstrate stable and efficient reinforcement post-training across stages.

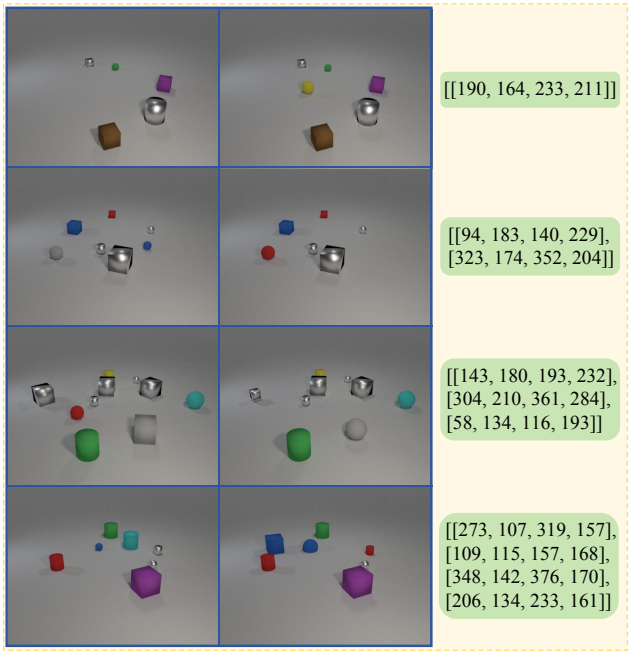


Figure 7. **Examples of synthesized Differential Grounding (DiG) scenes.** Each row shows an image pair with controlled visual differences in shape, color, size, or material, and the corresponding bounding boxes on the *"before"* image.

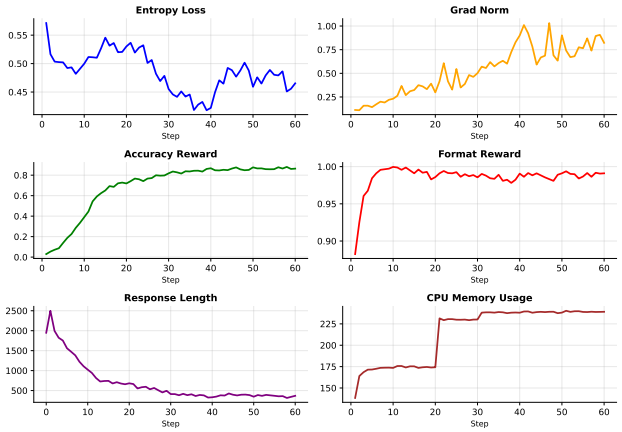


Figure 8. Training dynamics of the single-difference stage.

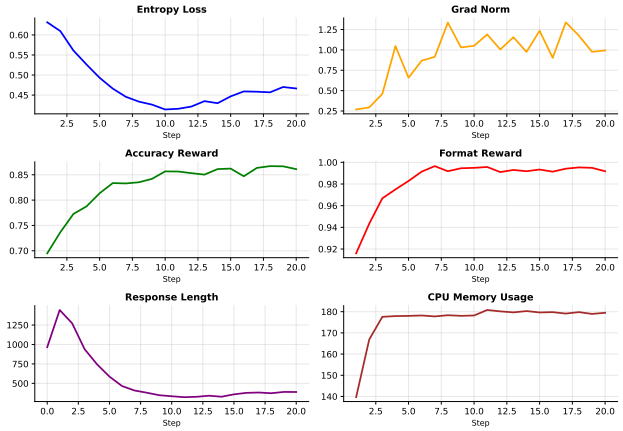


Figure 9. Training dynamics of the double-difference stage.

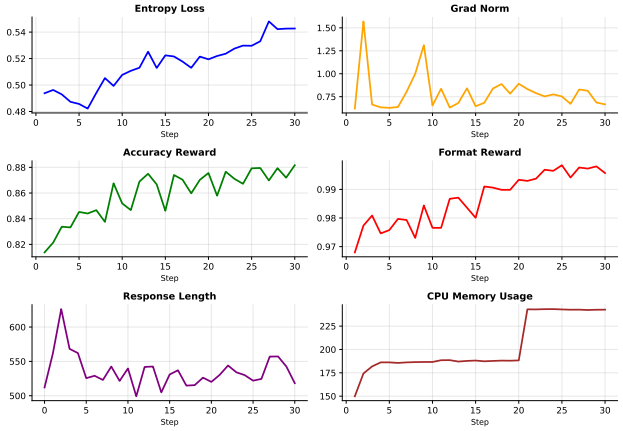
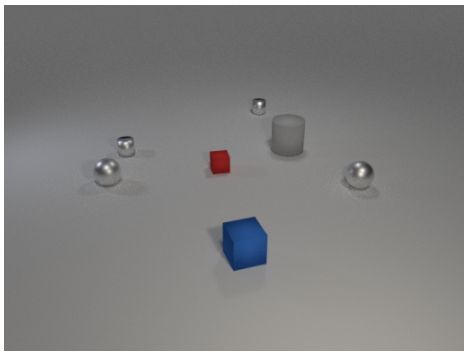


Figure 10. Training dynamics of the mix-difference stage.

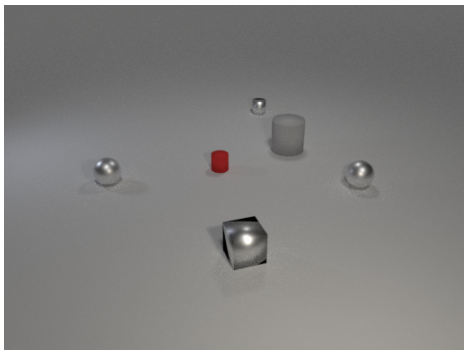
9. Case Study

To further illustrate the behavior of our model, we conduct a qualitative case study on the Differential Grounding (DiG) task, as shown in Fig. 11. Given a pair of images representing the "before" and "after" scenes, the model is required to identify and localize all visual differences between them. As demonstrated, the model accurately detects multiple types of changes, including object replacement and missing instances. Specifically, it recognizes that the red cube in the "before" image becomes a red cylinder, the blue cube is replaced by a metallic cube, and the small silver cylinder on the left disappears in the "after" scene. The predicted bounding boxes precisely correspond to these altered regions, confirming the model's ability to jointly reason over visual semantics and spatial grounding. These results highlight the model's fine-grained visual reasoning capacity and its robustness across complex difference configurations.

Case Study of Differential Grounding (DiG)



Before



After

Input: Example image pair from the DiG dataset. The model identifies and localizes visual differences between the two scenes.

Model Output:
Got it, let's compare the two images. First, the before image has a blue cube in the center, a red cube, and a small silver cylinder on the left. The after image has a metallic cube instead of the blue cube, and the red cube is now a red cylinder. Also, the small silver cylinder on the left is missing. Let's list the changes:
1. The red cube in the before image is now a red cylinder in the after. So the red cube's position is changed. 2. The blue cube in the before image is replaced by a metallic cube in the after. So the blue cube's position is changed. 3. The small silver cylinder (the one on the left) is missing in the after image. So that's a missing item.
Wait, need to check each object:
Before image objects: - Silver sphere left - Small silver cylinder (left) - Red cube (center) - Blue cube (bottom center) - Small silver cylinder (top) - Gray cylinder (right) - Silver sphere right
After image objects: - Silver sphere left (same) - Red cylinder (where red cube was) - Metallic cube (where blue cube was) - Gray cylinder (same) - Silver sphere right (same) - Small silver cylinder (top) same - Missing the small silver cylinder (left) and the blue cube.
So the changes are: - The red cube (before) is now a red cylinder (after), so the red cube's position is a change. - The blue cube (before) is replaced by a metallic cube (after), so that's a change. - The small silver cylinder on the left (before) is missing in after, so that's a missing item.
Therefore, the regions to box are the positions of the red cube, blue cube, and the small silver cylinder on the left. `</think>`
`<answer>`
[
[446, 415, 490, 487], // Red cube changed to red cylinder
[476, 613, 573, 767], // Blue cube replaced by metallic cube
[238, 374, 289, 447] // Small silver cylinder missing
]
`</answer>`

Figure 11. **Case study of Differential Grounding (DiG).** Illustration of input image pairs and corresponding model outputs highlighting localized visual discrepancies.